

# On Practical Accuracy of Edit Distance Approximation Algorithms

Hiroyuki Hanada <sup>\*†</sup>                      Mineichi Kudo <sup>\*</sup>  
 hana-hiro@live.jp                      mine@main.ist.hokudai.ac.jp  
 Atsuyoshi Nakamura <sup>\*</sup>  
 atsu@main.ist.hokudai.ac.jp

## Abstract

The edit distance is a basic string similarity measure used in many applications such as text mining, signal processing, bioinformatics, and so on. However, the computational cost can be a problem when we repeat many distance calculations as seen in real-life searching situations.

A promising solution to cope with the problem is to approximate the edit distance by another distance with a lower computational cost. There are, indeed, many distances have been proposed for approximating the edit distance. However, their approximation accuracies are evaluated only theoretically: many of them are evaluated only with big-oh (asymptotic) notations, and without experimental analysis. Therefore, it is beneficial to know their actual performance in real applications.

In this study we compared existing six approximation distances in two approaches: (i) we refined their theoretical approximation accuracy by calculating up to the constant coefficients, and (ii) we conducted some experiments, in one artificial and two real-life data sets, to reveal under which situations they perform best. As a result we obtained the following results: [Batu 2006] is the best theoretically and [Andoni 2010] experimentally. Theoretical considerations show that [Batu 2006] is the best if the string length  $n$  is large enough ( $n \geq 300$ ). [Andoni 2010] is experimentally the best for most data sets and theoretically the second best. [Bar-Yossef 2004], [Charikar 2006] and [Sokolov 2007], despite their middle-level theoretical performance, are

---

<sup>\*</sup>Graduate School of Information Science and Technology, Hokkaido University. Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido, 060-0814, Japan. (The affiliations of all authors at which the research is conducted, and current affiliation of Mineichi Kudo and Atsuyoshi Nakamura)

<sup>†</sup>Department of Computer Science, Nagoya Institute of Technology. Gokiso-cho, Showa-ku, Nagoya, Aichi, Japan. (Current affiliation of Hiroyuki Hanada)

experimentally as good as [Andoni 2010] for pairs of strings with large alphabet size.

**Keywords:** Edit Distance, Function Approximation, Distortion,  $q$ -gram

## 1 Introduction

The edit distance between two strings  $x$  and  $y$ , denoted by  $d_e(x, y)$  in this paper, is defined by the minimum number of character-wise edit operations (insertions, deletions or substitutions) to identify  $x$  and  $y$  (Section 2.1). The distance has been intensively researched because it naturally fits for many real-life situations: error detection in documents, noise analysis in signal processing, mutation-tolerant database searching in genomes and proteins, and so on [1, 2].

A weak point of the edit distance is its quadratic computation cost  $O(n^2)$ , where  $n$  is the string length to be compared. Many efforts, therefore, have been devoted to reduce the cost. They are separated by whether approximations of the distance are conducted or not. Unless some approximation is made, it is hard to reduce the worst-case computational cost from  $O(n^2)$ . Some methods without approximation [3, 4] achieve the worst-case computational time  $O(nk)$ , where  $k$  is the maximum edit distance to be considered. This means  $O(n)$  if  $k$  is a constant; but  $O(n^2)$  in the worst case because  $k$  can be  $n$ . Only approximation methods can achieve a linear or quasi-linear time such as  $O(n^{1+\varepsilon})$  or  $O(n(\log n)^m)$ . Then the next question with some approximation algorithms is whether they have sufficiently good approximation accuracy or not.

To answer the question, we will do in this paper the following studies:

**Theoretical evaluations:** We consider the *distortion* (Section 2.2.1) as a typical measure of approximation accuracy. Many approximation algorithms (four out of six in this paper) conducted only big-oh (asymptotic) analyses in the distortion, for example,  $O(n \log n)$  rather than  $100n \log n$ . However, in real-life situations, non-asymptotic distortions are desired. So we refine the analyses so as to reveal the constant factors.

**Experimental evaluations:** Most existing methods (all of six in this paper) have not received any experimental evaluation on the approximation accuracy. So we examine their experimental accuracy in three datasets (one artificial and two real).

## 2 Preparation

### 2.1 Definitions for strings

Throughout the paper, by  $\Sigma$  we denote the alphabet (the set of characters). Let  $\Sigma^n$  be the set of all strings of length  $n$ .

For a string  $x$ , we denote by  $|x|$  the length of  $x$ , by  $x[i]$  the  $i$ th character of  $x$ , and by  $x[i..j]$  the substring of  $x$  consisting of its  $i$ th to  $j$ th characters. A  $q$ -gram is a substring of length  $q$ .

The *edit distance* [1]  $d_e(x, y)$  for two strings  $x, y$  is defined by the minimum number of edit operations: inserting, deleting or substituting one character in  $x$  to make  $x$  be identical to  $y$ .

### 2.2 Distortion

#### 2.2.1 Definition

We use the *distortion*, also known as the *approximation factor*, as a measure of approximation accuracy of a function defined as follows:

**Definition 1** [5][6] *Given a set  $S$ , a non-negative function  $f(z)$  and a non-negative approximation function  $\tilde{f}(z)$ , the distortion of  $\tilde{f}(z)$  to  $f(z)$  is defined by the smallest  $K \in [1, +\infty)$  such that*

$$\exists K' \in (0, +\infty), \forall z \in S : f(z) \leq K' \tilde{f}(z) \leq K f(z).$$

The concept is illustrated in Fig. 1. Note that, in this paper,  $S$  is given as a set of pairs of strings  $\{z = (x, y)\}$  since we consider  $f(z) = d_e(x, y)$  and  $\tilde{f}(z) = \tilde{d}_e(x, y)$ , where  $\tilde{d}_e(x, y)$  is a string distance approximating  $d_e(x, y)$ . The value of  $K$  shows the ratio of the upper bound  $(K/K')f(z)$  to the lower bound  $(1/K')f(z)$ . A smaller value of distortion  $K$  ( $\geq 1$ ), therefore, means better approximation. Especially,  $K = 1$  means that  $f(z)$  and  $\tilde{f}(z)$  are proportional to each other.

#### 2.2.2 Asymptotic/non-asymptotic distortion analysis

The distortion is an intuitive measure for showing how close the value of the approximation distance  $\tilde{d}_e(x, y)$  is to the original distance  $d_e(x, y)$ . However, we have to pay attention to what the distortion actually means in several conditions (Fig. 2).

First we notice that the value of distortion, in general, becomes larger as the string length  $n$  increases, assuming  $|x| = |y| = n$  (Fig. 2, (a) and (b)). Taking this tendency into account, many of existing papers evaluate the distortions by big-oh notations, that is, how slowly the value  $K$  increases as  $n$  increases.

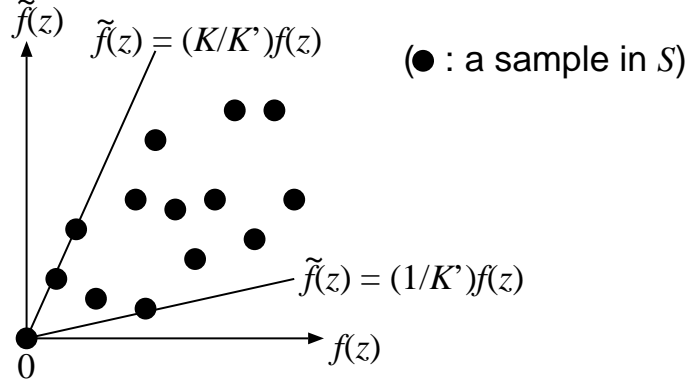


Figure 1: The concept of distortion  $K$  over a set  $S$

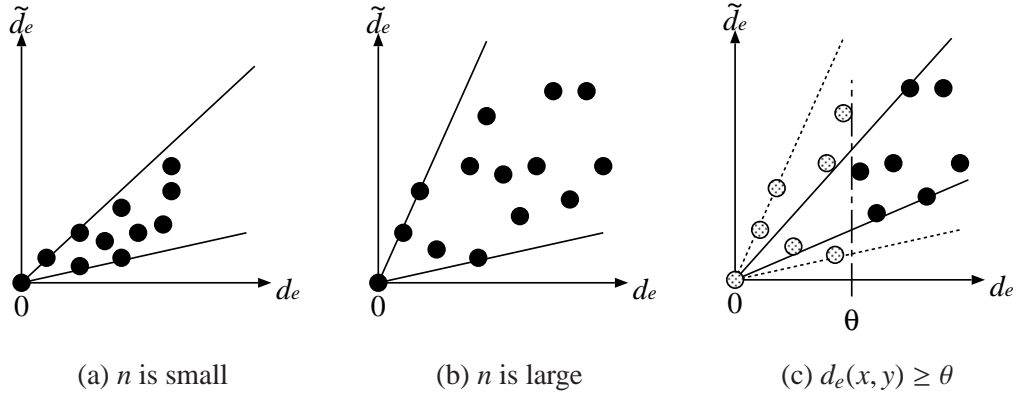


Figure 2: Several situations in distortion evaluation

We should also notice another tendency that the distortion is often affected strongly by string pairs with a small value of  $d_e$  (Fig. 2(b)(c)). To ignore such an exceptional situation, some of the existing methods are evaluated only in the range of  $d_e \geq \theta$  with a threshold  $\theta$  (Fig. 2(c)).

### 3 Outline of existing approximation methods

We chose six approximation algorithms to be compared from the two viewpoints: coverage of almost all state-of-the-art algorithms and implementation easiness. We explain those algorithms in four groups according to their characteristics.

**$q$ -gram-based algorithms** (two of: [7]=[Bar-Yossef 2004], [8]=[Sokolov 2007])

These two algorithms approximate the edit distance by counting occurrences of  $q$ -grams in given two strings, and then take the difference between them.

**Ulam-metric-based algorithms** (two of: [6]=[Charikar 2006], [9]=[Andoni 2009])

These two algorithms are originally developed for the *Ulam metric*, which is the edit distance in the set of strings whose characters are all distinct [6]. It can be shown that the Ulam metric is applicable for the edit distance between general strings with some simple operations (Section 5.1). The distance computation of the two algorithms exploits the property that every string does not contain the same character twice or more. For example, in [Charikar 2006], the distance is defined as the sum of  $|1/(x^{-1}[b] - x^{-1}[a]) - 1/(y^{-1}[b] - y^{-1}[a])|$  for all pairs  $(a, b) \in \Sigma \times \Sigma$ , where  $x^{-1}[a]$  denotes the position of  $a$  found in the string  $x$  (omitted if  $a$  is not in  $x$ ).

**Restricted alignment algorithms** (one of: [10]=[Andoni 2010])

The edit distance can be regarded as a character-wise *alignment* between two strings [1]. [Andoni 2010] uses  $q$ -gram-wise alignment instead and assures certain approximation accuracy even if a pruning in the calculation is conducted<sup>1</sup>.

**Shrinking algorithms** (one of: [11]=[Batu 2006])

Batu’s algorithm converts given strings into shorter ones by merging some characters into one such as “abcbababc”  $\rightarrow$  “XYX” with the rule “abc”  $\rightarrow$  “X” and “bb”  $\rightarrow$  “Y”. Then it computes the edit distance of the converted strings as the approximated distance.

## 4 Refined theoretical distortions

### 4.1 Outline

We re-analyzed the six algorithms to obtain their distortions with constant factors. The results are shown in Table 1.

Before analyzing the table in detail in Section 4.3, in Section 4.2 we explain how the constant factors are extracted from big-oh notations, and how the accuracy evaluations with inequalities are converted to distortions with a threshold  $\theta$ .

---

<sup>1</sup>The algorithm of [Andoni 2010] needs  $O(n^2)$  time if no pruning is made, which is equal to that of the edit distance.

Table 1: Refined distortions. Here,  $\tilde{d}_e$  is the approximated distance of  $d_e$ ; the strings are limited to length  $n$ ; a threshold  $\theta$  is employed to limit  $d_e \geq \theta$  in some algorithms. In logarithms, the bases are 2 for lg and  $e$  for ln, respectively.

Algorithm	Original distortion	Original inequality	Refined distortion
[Bar-Yossef 2004] [7]		$\begin{cases} d_e \leq k \Rightarrow \tilde{d}_e \leq 4kq,^\dagger \\ d_e \geq 13(kn)^{\frac{2}{3}} \Rightarrow \tilde{d}_e \geq 8kq \end{cases}$	$\frac{13}{2\theta^{1/3}} n^{2/3}$
[Batu 2006] [11]	$\min \left\{ n^{\frac{1}{3}+o(1)}, (d_e)^{\frac{1}{2}+o(1)} \right\}$		$4(2c-1) \left( \lg((2c-3)k) + 1 + \frac{(c-1)^2}{c} \right)^\ddagger$
[Charikar 2006] [6]	$O(n \log n)^{\dagger\dagger}$		$48n(1 + \ln n) / \max\{1, \theta\}$
[Sokolov 2007] [8]		$\begin{cases} d_e \leq k \Rightarrow \tilde{d}_e \leq \frac{2k(n+2)}{n}, \\ d_e > k \Rightarrow \tilde{d}_e \geq \frac{2k-8}{n} \end{cases}$	$\begin{cases} +\infty & (\theta \leq 5), \\ \frac{n\theta+2}{\theta-5} & (\theta > 5) \end{cases}$
[Andoni 2009] [9]	$O(n)^{\dagger\dagger}$		$3400n$
[Andoni 2010] [10]	$12 \lg n^{\ddagger\ddagger}$		$12 \lg n^{\ddagger\ddagger}$

Note:

$^\dagger$   $q$  denotes the  $q$ -gram. In the algorithm,  $q$  is set to  $n^{2/3}/(2k^{1/3})$ .

$^\ddagger$   $c = \max\{(\lg \lg n)/(\lg \lg \lg n), 2\}$ .

$^{\dagger\dagger}$  In [Charikar 2006] and [Andoni 2009], the distortions are derived for the Ulam metric as  $O(\log n)$  and  $O(1)$ , respectively. We multiplied them by  $O(n)$  (more precisely,  $2n$ ) so as to be applicable to general strings (Section 5.1).

$^{\ddagger\ddagger}$  The distortion is shown in the original paper ([10], pp. 16 in the full version).

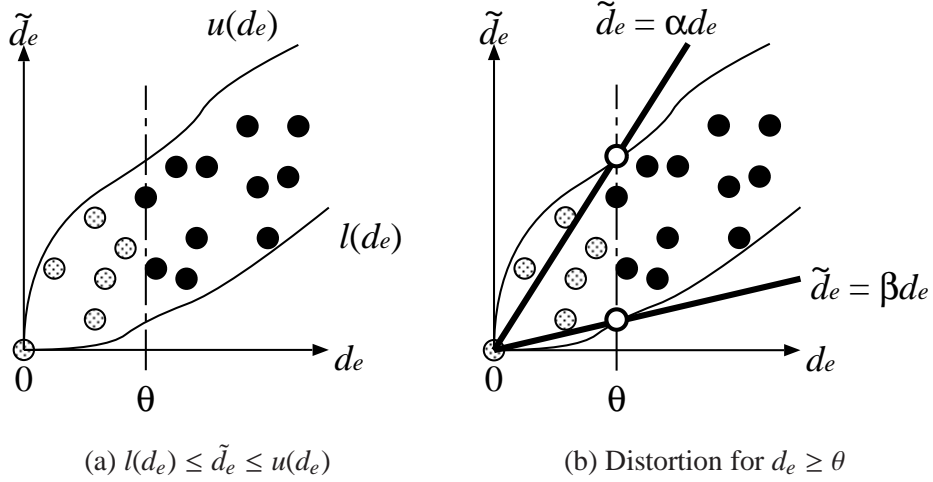


Figure 3: Conversion of lower and upper bounds to a distortion

## 4.2 Derivation of distortions

For each algorithm whose distortion is given in a big-oh notation ([Batu 2006], [Charikar 2006], [Andoni 2009] and [Andoni 2010]), we examined every step in the algorithm. The detailed derivations are given in Appendix A.

For each algorithm whose accuracy is bounded by inequalities ([Bar-Yossef 2004] and [Sokolov 2007]), we calculated its distortion by the following procedure. Detailed distortion calculations for the two algorithms are shown in Appendix B.

Let  $\tilde{d}_e$  be bounded by two functions of  $d_e$  as  $l(d_e) \leq \tilde{d}_e \leq u(d_e)$  for  $d_e \geq \theta$  (Fig. 3(a)). Then the distortion  $K$  of  $\tilde{d}_e$  for  $d_e \geq \theta$  is upper-bounded by  $K_\theta = u(\theta)/l(\theta)$  under the monotonicity of slopes  $u(d_e)/d_e$  and  $l(d_e)/d_e$ . Indeed, if  $u(d_e)/d_e$  and  $l(d_e)/d_e$  are monotonically decreasing and increasing in  $d_e \geq \theta$ , respectively, then  $K = (\sup_{d_e \geq \theta} u(d_e)/d_e) / (\inf_{d_e \geq \theta} l(d_e)/d_e) \leq (u(\theta)/\theta) / (l(\theta)/\theta) = u(\theta)/l(\theta) = K_\theta$ . Therefore we can obtain the distortion when the monotonicity of them are confirmed.

## 4.3 Comparison of calculated distortions

Now we examine the refined distortions shown in Table 1. We note that all these algorithms can be now compared in a unified expression.

First we classify these algorithms in the complexity order. Note that we can assume that  $\theta$  takes an order between  $O(1)$  and  $O(n)$  since the edit distance takes a value between 0 and  $n$ . Assuming  $\theta = O(1)$  as an ordinary case, they are ordered

as:

- Sub-logarithmic ( $O((\log \log n)^2)$ ): [Batu 2006]
- Logarithmic ( $O(\log n)$ ): [Andoni 2010]
- Sublinear ( $O(n^{\alpha(<1)})$ ): [Bar-Yossef 2004]
- Linear ( $O(n)$ ): [Sokolov 2007], [Andoni 2009]
- Super-linear ( $O(n \log n)$ ): [Charikar 2006]

Therefore, [Batu 2006] is the best for  $\theta = O(1)$  then [Andoni 2010] follows. For  $\theta = O(n)$ , [Charikar 2006] also has the same logarithmic order. Thus [Charikar 2006] and [Andoni 2010] are comparable for  $\theta = O(n)$ .

Next let us compare the distortions in more detail. Since the refined distortions reveal the constants, we can compare algorithms for every specific value of  $n$ . We show the result in Fig. 4. In the figure we set  $\theta = n$  (maximum  $\theta$ ) for [Bar-Yossef 2004], [Charikar 2006] and [Sokolov 2007] to evaluate optimistic distortion values. It is observed as expected that [Batu 2006] outperforms the others if  $n$  is large enough. However, when  $n$  is not so large, say,  $n \leq 300$ , [Bar-Yossef 2004] is the best. Such a range of effective  $n$  is not obtained until our analyses made clear the constant factors.

Focusing on the absolute value of distortion, it ranges from 10 to 100 for  $100 \leq n \leq 10000$ . We might need to investigate whether such large values are acceptable in real-life applications, keeping in mind that they are evaluated in the worst case.

## 5 Experimental comparison

### 5.1 Procedure

Next we compared them experimentally to know their practical usefulness.

For each data set that will be explained in detail later, we make ready a set  $S$  of 10,000 pairs of strings  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_{10000}, y_{10000})\}$ . We computed the distortion for  $S$  for the six approximation distances.

We used one artificial and two real-life data sets as follows:

**Random** ( $n \in \{100, 300, 1000\}$ ,  $|\Sigma| \in \{4, 20\}$ ,  $e \in \{4, 30\}$ ):

First we choose  $x$  from  $\Sigma^n$  at random with equal probability and initialize  $y$  by  $x$ . Then we modify  $y$  until the total operation cost becomes  $e$ : (a) replace a randomly chosen character in  $y$  with a randomly chosen character from  $\Sigma$  (probability:  $2/3$ , cost: 1) or (b) delete a randomly chosen character in  $y$



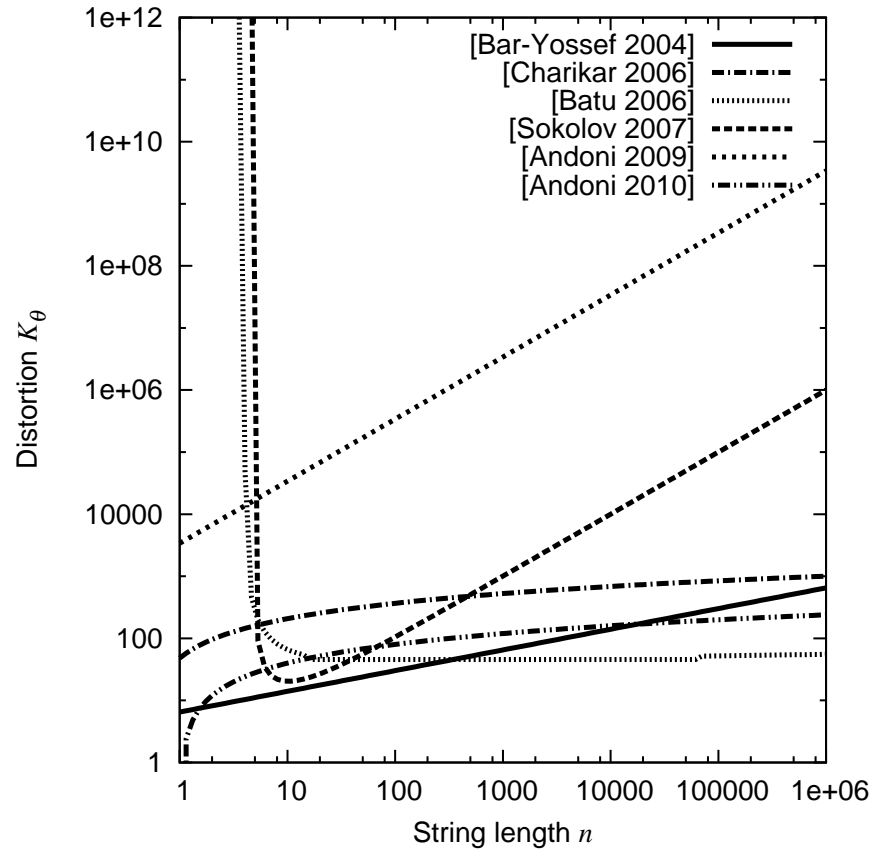


Figure 4: Distortions of six approximation methods.  
Note:  $\theta = n$  for [Bar-Yossef 2004], [Charikar 2006] and [Sokolov 2007].

and then insert a randomly chosen character at a randomly chosen position (probability:  $1/3$ , cost:  $2$ ), where all random choices of characters and positions are conducted with equal probability. Note that  $d_e(x, y)$  equals  $e$  in most cases but can be less than  $e$ .

**DDBJ** ( $n \in \{100, 300, 1000\}$ ):

DDBJ (DNA Data Bank of Japan) is a DNA nucleobase sequence database service [12]. We used “ddbjhum1” data ( $|\Sigma| = 15$ ; 4 of them occupy 99.95%). To unify the string length in each data set, we constructed the data set as follows: For  $n = 100$ , we gathered strings of length 100 to 299 in ddbjhum1 and truncated the 101st character or after. Similarly, for  $n = 300$  and  $n = 1000$ , we collected strings of length 300 to 999 for  $n = 300$  and 1000 to 2999 for  $n = 1000$ , respectively.

**UniProt** ( $n \in \{100, 300, 1000\}$ ):

UniProt (Universal Protein Resource) is an amino acid sequence (i.e. protein) database service [13]. We used “UniProtKB-SwissProt” data ( $|\Sigma| = 25$ ; 20 of them occupy 99.99%). We conducted the data set constructions in the same manner as in DDBJ.

For the algorithms assuming the Ulam metric ([Charikar 2006] and [Andoni 2009], Section 3), where all characters in a string are expected to be distinct, we “expanded” the alphabet from  $\Sigma$  to  $\Sigma^t$  for each string pair  $x, y$  so that  $(x[1..t], \dots, x[n-t+1..n])$  are distinct and so do  $(y[1..t], \dots, y[n-t+1..n])$  with as small  $t$  as possible. It can be shown that the distortion with this expansion is at most  $2t$  times that under the Ulam metric [6].

When algorithms have parameters ([Bar-Yossef 2004], [Batu 2006], [Sokolov 2007] and [Andoni 2010]), we chose the smallest distortions over some candidates of parameters as follows:

- $q \in \{2, 4, 6\}$  for  $q$ -grams ([Bar-Yossef 2004] and [Sokolov 2007]<sup>2</sup>).
- $c \in \{2, 4\}$  and  $j = 1$  for [Batu 2006] (see Appendix A for details). As a result, the theoretical distortion of [Batu 2006] is  $(2c - 1) \cdot [4c + \{8(2c - 3)k\}^{c-1}] / c = 12[1 + \lceil \lg |\Sigma| \rceil] = 72$  with  $c = 2$  and  $|\Sigma| = 20$ , a constant against  $n$ . It needs  $O(n^2)$  time.
- Tree node pruning (the trade-off between the computational time and the accuracy) is not conducted on [Andoni 2010] (the highest accuracy). It needs  $\Omega(n^2)$  time.

---

<sup>2</sup>Following the description in the papers [Bar-Yossef 2004] and [Sokolov 2007],  $B$  and  $q_1$  corresponds to  $q$ , respectively. In [Sokolov 2007], parameter  $q_2$  is also set to  $q$ .

Table 2: Best algorithm according to the alphabet size  $|\Sigma|$ , the string length  $n$  and the number of edits (an upper bound of the edit distance)  $e$ .

Edit distance	$ \Sigma  = 4$			$ \Sigma  = 20$		
	$n = 100$	$n = 300$	$n = 1000$	$n = 100$	$n = 300$	$n = 1000$
$e = 4$	Andoni 2010			Sokolov 2007	Charikar 2006	
$e = 30$	Sokolov 2007	Charikar 2006			Bar-Yossef 2004	
$e \sim n$ (real-life)	Andoni 2010					

## 5.2 Results

We show the experimental results in Fig. 5, Fig. 6 and Table 2. From Fig. 5 we see that actual values of distortion are far less than their theoretical values, often 10 times or more (one scale mark in Fig. 5). This is mainly because theoretical distortions are obtained in the worst case but real data are not the case.

We also see from Fig. 5 that the behavior (the outline of curves) obeys well the theoretical prediction, especially in [Batu 2006] and [Andoni 2010], whose asymptotic distortions are  $O(1)$  and  $O(\log n)$  under the condition of this experiment, respectively.

Then we list the best algorithms depending on  $|\Sigma|$ ,  $n$  and  $e$  in Table 2 and the detailed comparison in Fig. 6. We assumed “ $e \sim n$ ” in the two real-life data sets (DDBJ and UniProt) in Table 2, since they contain strings coming from many organic components and thus most string pairs have large (nearly  $n$ ) edit distance.

We can see that [Andoni 2010], theoretically the second best, is almost always the best: it is the best for the two real-life data sets (DDBJ and UniProt) and nearly the best even for Random data set. On the other hand, theoretically the best algorithm [Batu 2006] did not yield the smallest distortion for any data set. Rather, as seen in Table 2, [Bar-Yossef 2004], [Charikar 2006] or [Sokolov 2007] becomes the best for Random data sets. Indeed, from Fig. 6, the conditions under which these algorithms achieved the smallest or near distortion are  $|\Sigma| = 20$  for [Bar-Yossef 2004] and [Sokolov 2007], and  $e = 4, 30$  for [Charikar 2006]. The possible explanation of their good achievements is as follows:

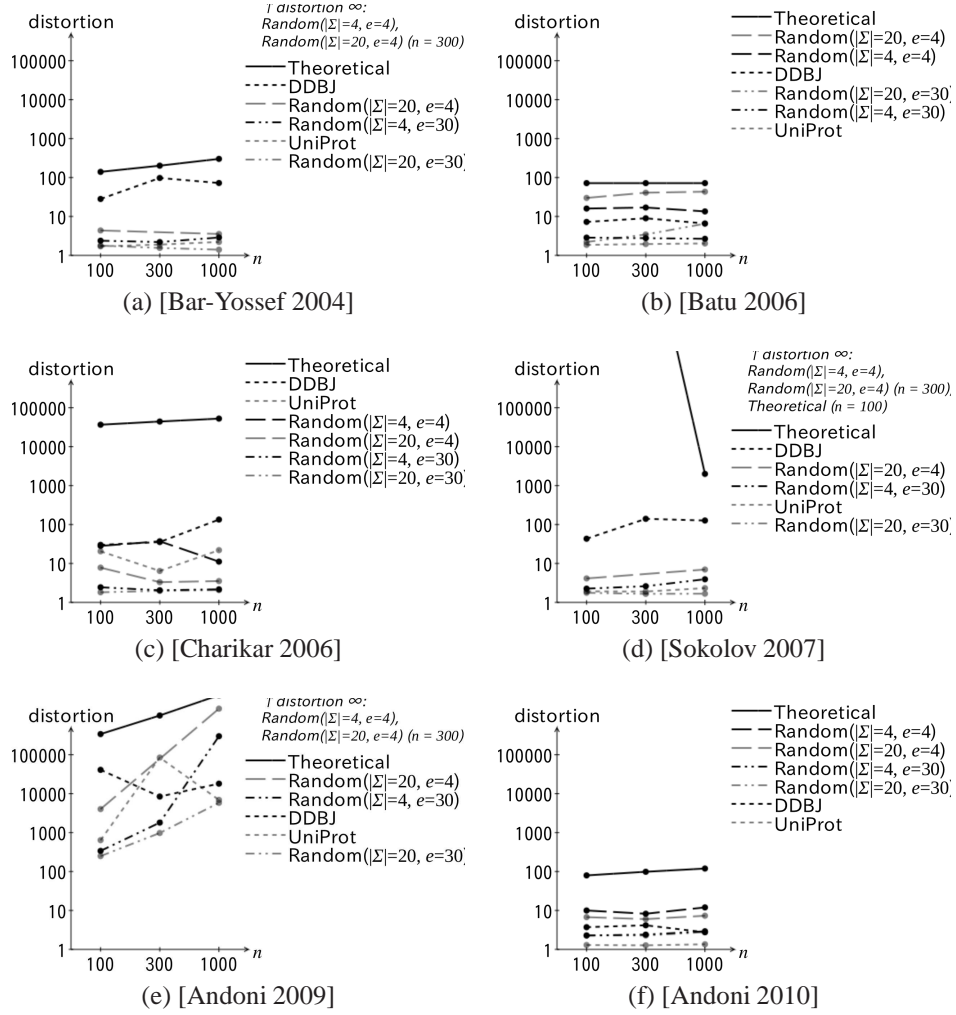


Figure 5: Experimental distortions of six algorithms. Gray lines denote  $|\Sigma| = 20$  data sets including UniProt. The theoretical value of [Batu 2006] is different from that in Table 1 (constant against  $n$ ; see Section 5.1).

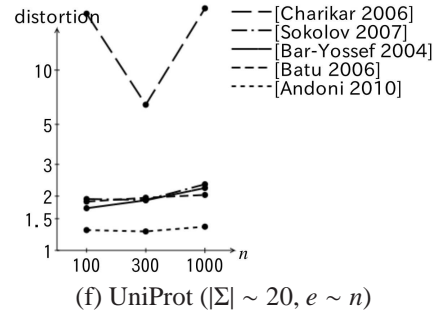
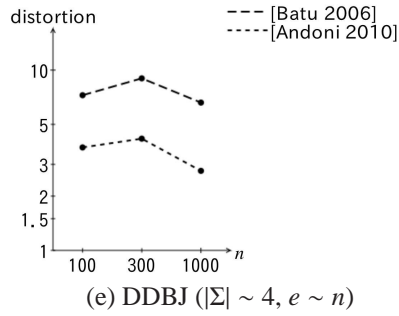
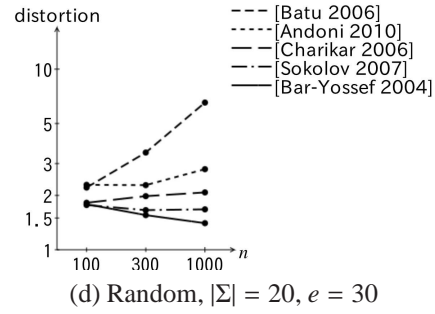
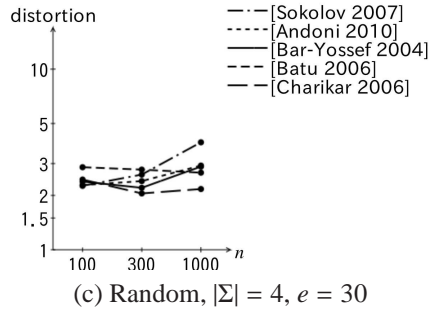
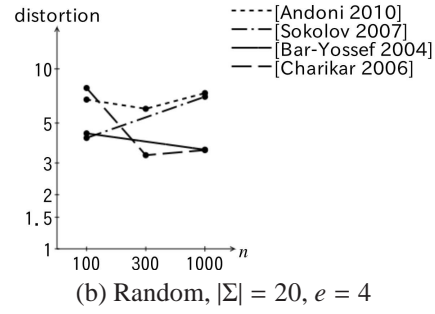
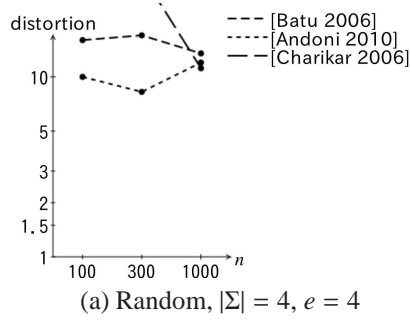


Figure 6: Distortions of six data sets. Distortions larger than 30 are omitted from the charts.

- [Bar-Yossef 2004] and [Sokolov 2007] showed better results for relatively large  $|\Sigma|$ . This is because they are  $q$ -gram-based algorithms. When  $|\Sigma|$  is large,  $q$ -grams over  $\Sigma$  appearing in a string become more distinct even if the value of  $q$  is small. This means that the effect of appearance order<sup>3</sup> disappears and thus  $q$ -gram distance becomes close to the edit distance.
- [Charikar 2006] showed better results for  $|\Sigma| = 20$  or ( $|\Sigma| = 4$  and  $e = 30$ ). This is because the distortion due to the alphabet expansion (Section 5.1) can be small. When  $|\Sigma|$  is large or  $e$  is not so small compared with  $n$ , the expansion length  $t$  to satisfy the Ulam condition can be small, especially in Random data set because uniform randomness works well.

We have analyzed only the distortion so far. However, there is a trade-off between the distortion and the computational cost. The computational costs of the six algorithms ranges from  $O(n)$  ([Bar-Yossef 2004], [Charikar 2006] and [Sokolov 2007]) to  $O(n^{1+\varepsilon})$  ([Batu 2006], [Andoni 2009] and [Andoni 2010]). In addition, in the latter three algorithms, we can control the trade-off by changing the value of  $\varepsilon$ . Since we carried out the experiment with  $\varepsilon \sim 1$  (i.e. the least distortion at the expense of large time complexity  $O(n^2)$  same as the edit distance), it might be better to take into account the time complexity for choosing an algorithm in practical problems.

## 6 Conclusion

We have compared six approximation algorithms of the edit distance in distortion, a measure of approximation accuracy, from the practical point of views: theoretical distortions without big-oh (asymptotic) notations, and experimental distortions in artificial and real-life data.

By the theoretical comparison, we have revealed the conditions on the string length  $n$  for which these algorithms work best. The asymptotically best algorithm, [Batu 2006], was practically the best for  $n \geq 300$ , while [Bar-Yossef 2004] was the best for smaller  $n$ . In the experimental comparison, however, [Batu 2006] did not yield the best distortion for any data set, while [Andoni 2010] was the best or nearly best for most of real data sets, and [Bar-Yossef 2004], [Charikar 2006] and [Sokolov 2007] were the best or nearly best for large  $|\Sigma|$ . Since they are faster than [Batu 2006] and [Andoni 2010], it is worth changing the algorithm depending on the problems at hand.

---

<sup>3</sup>A counter example is  $x = \text{"abcdefgh"}$  and  $y = \text{"efghabcd"}$ : the difference of appearance order makes the edit distance be large ( $d_e(x, y) = 8 = n$ ) while 2-gram distance [8] is small (2).

The contribution of the paper is that this analysis revealed the ranges of  $n$  where each approximation algorithm works better than the others with the absolute value of distortion, and that the experimental results revealed a large gap between theoretical and practical values of distortion in the algorithms.

In the future work, in addition to the discussion on the computational cost, we will narrow the gap between theoretical and experimental distortions by controlling  $d_e$  and  $\theta$  in more detail (Section 4.3 and 5.2). We are also planning to apply them for real-life applications like biological sequence analyses, signal processing, or logging data analyses to confirm the accuracy and the computational time are practical enough.

## Appendix

### A Details of the distortion refinement without the big-oh notation

Let  $\log_b^* x$ , called the *iterated logarithm* [14], be the minimum  $i \geq 0$  such that  $\underbrace{\log_b(\log_b(\dots \log_b x))}_{i \text{ 'log's'}} \leq 1$ . If  $x \leq 1$  then  $\log_b^* x \stackrel{\text{def}}{=} 0$ .  $\log_b^* x$  grows very slowly compared to  $x$ , e.g.  $\lg^* x = 3$  if  $x \in (4, 16]$  and  $\lg^* x = 4$  if  $x \in (16, 65536]$ .

#### A.1 [Batu 2006]

In Batu's algorithm [11], we first divide a string  $x$  into blocks of length  $c$  to  $2c - 1$  and compute the edit distance block-wise (i.e. treating a block as a character). As a result, the computational cost becomes  $O((n/c)^2)$  after one division. The algorithm has two parameters  $c \geq 2$ ,  $j \geq 1$ .<sup>4</sup>  $j$  describes the number of the *alphabet reductions* (a string conversion process that only determines the boundaries of blocks). Note that we need to increase  $c$  in accord with  $n$  by  $c = \omega(1)$  to assure  $o(n^2)$ -time computation. The authors of the paper take  $c = (\lg \lg n) / \lg \lg \lg n$  (the end of Section 5 of [11]). In Section 4 we took  $c = \max\{\lg \lg c / (\lg \lg \lg c), 2\}$  instead. In Section 5 we fixed  $c = 2$  for the theoretical distortion since we took only  $c = 2, 4$  for the experiment.

---

<sup>4</sup>There is another parameter  $\ell$ , but we fixed  $\ell = 1$  since it is enough for the single use of the distance ([11], pp. 799).

The distortion  $K$  is given by

$$K = (2c - 1) \cdot O((3c^2 \log c)^c / c + \log^* kc) \quad (1)$$

(Theorem 4.1 in [11], pp. 797)

$$= (2c - 1) \cdot [4c(\log^* kc + O(1)) + O((3c^2 \log c)^c)]/c$$

(Lemma 4.5 in [11], pp. 797)

$$= (2c - 1) \cdot [4cj + O((3c^2 \log c)^c)]/c. \quad (2)$$

(Lemma 4.5 in [11], pp. 797)

where  $k = \lceil \lg |\Sigma| \rceil$  is the number of bits to describe a character. The remained big-oh notation  $O((3c^2 \log c)^c)$  is evaluated as follows:  $O((3c^2 \log c)^c)$  is obtained from  $2^{k_j}$  where  $k_i = (c - 1) \cdot (\lceil \lg((2c - 3)k_{i-1}) \rceil + 2)$ ,  $k_0 = k$  (pp. 796 in [11]).

#### A.1.1 The case of $j = 1$

If  $j = 1$ , used in Section 5, then  $k_1 = (c - 1) \cdot (\lceil \lg((2c - 3)k_{i-1}) \rceil + 2) \leq (c - 1) \cdot (\lg((2c - 3)k) + 3)$  and thus the distortion becomes

$$K \leq (2c - 1) \cdot [4c + \{8(2c - 3)k\}^{c-1}]/c. \quad (3)$$

#### A.1.2 The case $j$ is large enough

Then we consider the case  $j$  is large enough for the small distortion. In this case  $k_j$  becomes the fixed point of  $k_i = (c - 1) \cdot (\lceil \lg((2c - 3)k_{i-1}) \rceil + 2)$ . We can easily confirm that  $k_j \leq 4(c - 1)^2$  since it is larger than  $(c - 1) \cdot (\lceil \lg((2c - 3)k) \rceil + 2)$  for any  $c \geq 2$ .<sup>5</sup> In addition,  $j$  is large enough with  $\lg((2c - 3)k) + 1$  if  $k \geq k_j$  since the number of binary digits of  $k_i$  in the recurrence is reduced by at least one except for the final recurrence. As a result, from the expression (2), an upper bound of the distortion becomes

$$K = (2c - 1) \cdot [4cj + O((3c^2 \log c)^c)]/c$$

$$\leq 4(2c - 1) \left( \lg((2c - 3)k) + 1 + \frac{(c - 1)^2}{c} \right).$$

---

<sup>5</sup>We found an upper bound  $\hat{k}_j = 4(c - 1)^2$  as follows: since  $k$  is asymptotically larger than  $(c - 1) \cdot (\lceil \lg((2c - 3)k) \rceil + 2)$  in  $k$ ,  $\hat{k}_j$  must satisfy  $\hat{k}_j \geq (c - 1) \cdot (\lceil \lg((2c - 3)\hat{k}_j) \rceil + 2)$ . As a result,  $k_j = \omega(c)$  is required. Thus we first take  $k = \gamma(c - 1)^2$  and then supplied the constant  $\gamma$  to satisfy the inequality.



## A.2 [Charikar 2006]

The distortion of Charikar's method [6] is evaluated as  $O(\log n)$  for Ulam metric. First we show its value without big-oh notation. The approximation function  $\|f(P) - f(Q)\|$ , where  $P$  and  $Q$  are strings satisfying the Ulam condition, is evaluated as follows in [6]:

$$\begin{aligned} \|f(P) - f(Q)\| &\leq 3(1 + \ln n) \leq 3(1 + \ln n) \frac{d_e(P, Q)}{\theta} \\ &\quad (\text{if } P \neq Q; \text{ in Lemma 2.2, pp.211 in [6]}) \\ \|f(P) - f(Q)\| &\geq d_e(P, Q)/8 \\ &\quad (\text{in Lemma 2.3, pp.212 in [6]}) \end{aligned}$$

Thus we get  $d_e(P, Q)/8 \leq \|f(P) - f(Q)\| \leq 3(1 + \ln n) \frac{d_e(P, Q)}{\max\{1, \theta\}}$ , where  $\theta$  is replaced with  $\max\{1, \theta\}$  since the expression above does not consider the case  $d_e(P, Q) = 0$ . This concludes the distortion of  $\|f(P) - f(Q)\|$  for the Ulam metric is  $\frac{24(1+\ln n)}{\max\{1, \theta\}}$ .

In addition, in the manner in Section 5.1, the distortion for any strings is  $\frac{24(1+\ln n)}{\max\{1, \theta\}} \cdot 2n = \frac{48n(1+\ln n)}{\max\{1, \theta\}}$  since  $t$  is at most  $n$ .

## A.3 [Andoni 2009]

The distortion for [Andoni 2009] [9] is concluded as  $O(1)$  for the Ulam metric. We have removed the big-oh notation as follows: The approximation function  $d_{\text{NEG}, \infty, 1}(\phi(P), \phi(Q))$ , where  $P$  and  $Q$  are strings satisfying the Ulam condition, is evaluated as follows in [9]:

$$\begin{aligned} d_{\text{NEG}, \infty, 1}(\phi(P), \phi(Q)) &\geq \underline{d_e}(P, Q)/50 \\ &\quad (\text{Proof of Theorem 1.1, pp.870}) \\ d_{\text{NEG}, \infty, 1}(\phi(P), \phi(Q)) &\leq 17\underline{d_e}(P, Q) \\ &\quad (\text{Proof of Theorem 1.1, pp.871}) \\ \underline{d_e}(P, Q) \leq d_e(P, Q) &\leq 2\underline{d_e}(P, Q) \\ &\quad (\text{Section 1.5, pp.868}) \end{aligned}$$

As a result, the distortion for Ulam metric is calculated as  $50 \cdot 17 \cdot 2 = 1700$ . In addition, in the manner in Section 5.1, the distortion for any strings is  $1700 \cdot 2n = 3400n$  since  $t$  is at most  $n$ .

## B Details of the distortion calculation from inequalities

### B.1 [Bar-Yossef 2004]

The upper and the lower bounds of [7]=[Bar-Yossef 2004] are given by

$$\begin{cases} d_e \leq k \Rightarrow \tilde{d}_e \leq 4kq, \\ d_e \geq 13(kn)^{\frac{2}{3}} \Rightarrow \tilde{d}_e \geq 8kq. \end{cases} \quad (\text{with } q = n^{2/3}/(2k^{1/3}))$$

As a result we obtain

$$\frac{4}{13}d_e \leq \tilde{d}_e \leq 2(d_en)^{2/3}.$$

As shown in Section 4.2, since  $u(d_e)/d_e = 2(n^2/d_e)^{1/3}$  and  $l(d_e)/d_e = 4/13$  are monotonically decreasing and increasing, respectively, the distortion for  $d_e \geq \theta$  is  $K_\theta = u(\theta)/l(\theta) = 2(\theta n)^{2/3}/(\frac{4}{13}\theta) = (13n^{2/3})/(2\theta^{1/3})$ .

### B.2 [Sokolov 2007]

The upper and the lower bounds of [8]=[Sokolov 2007] are given by

$$\begin{aligned} d_e(x, y) \leq k &\Rightarrow \tilde{d}_e(x, y) \leq (2k(n+2))/n, \\ d_e(x, y) > k &\Rightarrow \tilde{d}_e(x, y) \geq 2(k-4)/n. \end{aligned} \quad (4)$$

Note that the distortion should be treated as  $+\infty$  if  $\theta \leq 5$  since  $\tilde{d}_e(x, y)$  can be zero if  $d_e(x, y)$  is less than 5, that is,  $k$  is less than 4, from (4). Otherwise we obtain

$$2(d_e - 5)/n \leq \tilde{d}_e \leq (2d_e(n+2))/n.$$

As shown in Section 4.2, since  $u(d_e)/d_e = (2d_e(n+2))/(nd_e)$  and  $l(d_e)/d_e = 2(d_e - 5)/(nd_e)$  are monotonically decreasing and increasing, respectively, the distortion for  $d_e \geq \theta$  is  $K_\theta = u(\theta)/l(\theta) = [(2\theta(n+2))/n]/[2(\theta - 5)/n] = (\theta(n+2))/(\theta - 5)$ .

## References

- [1] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.
- [2] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.
- [3] G. M. Landau and U. Vishkin. Fast parallel and serial approximate string matching. *Journal of Algorithms*, 10(2):157–169, 1989.

- [4] E. Myers. A sublinear algorithm for approximate keyword searching. *Algorithmica*, 12(4-5):345–374, 1994.
- [5] J. Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, 2002.
- [6] M. Charikar and R. Krauthgamer. Embedding the Ulam metric into  $l_1$ . *Theory of Computing*, 2(11):207–224, 2006.
- [7] Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, and R. Kumar. Approximating edit distance efficiently. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 550–559, 2004.
- [8] A. M. Sokolov. Vector representations for efficient comparison and search for similar strings. *Cybernetics and Systems Analysis*, 43(4):484–498, 2007.
- [9] A. Andoni, P. Indyk, and R. Krauthgamer. Overcoming the  $l_1$  non-embeddability barrier: Algorithms for product metrics. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 865–874, 2009.
- [10] A. Andoni, R. Krauthgamer, and K. Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the 51st IEEE Annual Symposium on Foundations of Computer Science*, pages 377–386, 2010. Full version available at <http://arxiv.org/abs/1005.4033>.
- [11] T. Batu, F. Ergun, and C. Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 792–801, 2006.
- [12] Hideaki Sugawara, Kazuho Ikeo, Satoshi Fukuchi, Takashi Gojobori, and Yoshio Tateno. DDBJ dealing with mass data produced by the second generation sequencer. *Nucleic Acids Research*, 37:Database issue D16–D18, 2009. <http://www.ddbj.nig.ac.jp/index-e.html>.
- [13] UniProt Consortium. The universal protein resource (uniprot). *Nucleic Acids Research*, 37:D169–D174, 2009. <http://www.pir.uniprot.org/>.
- [14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.